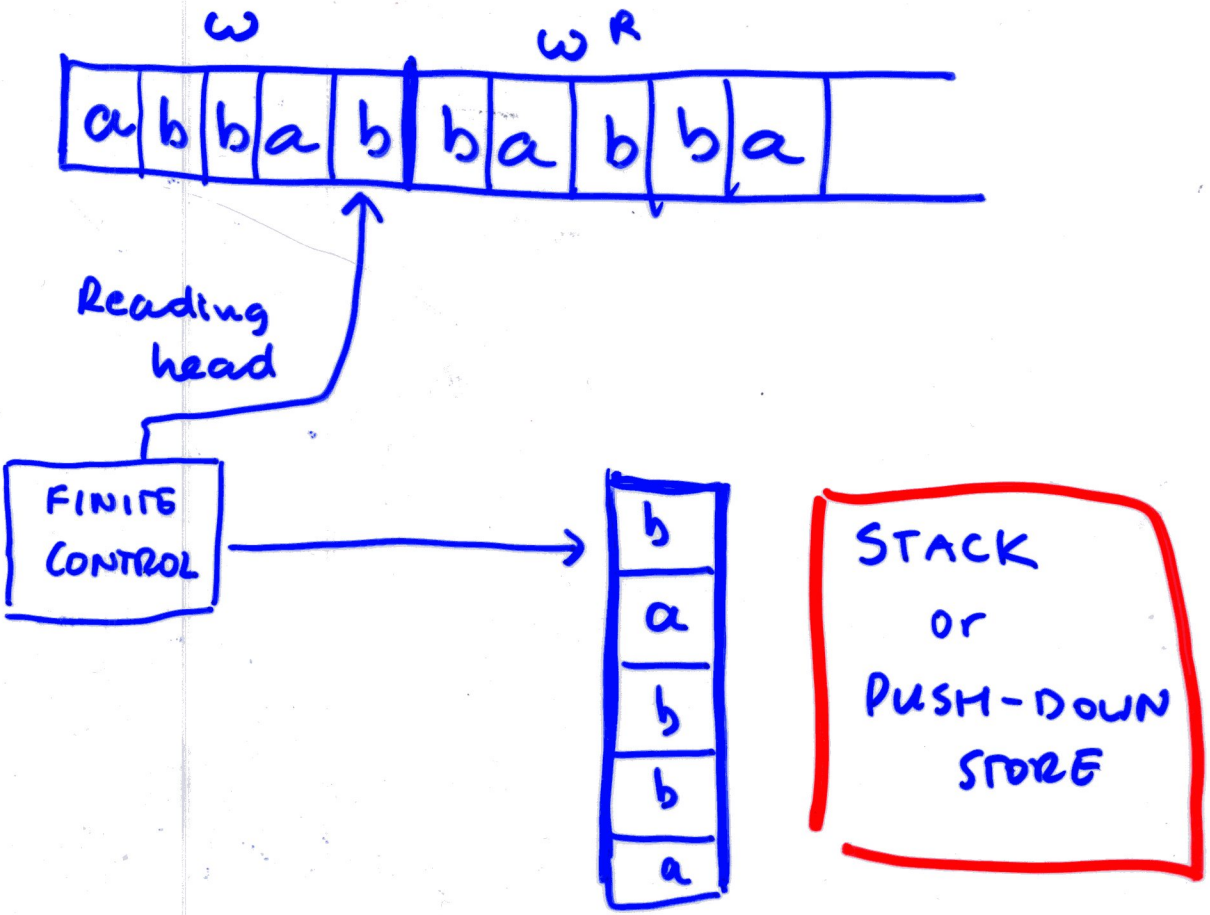
PUSH-DOWN AUTOMATA

Picture

$L = \{ w w^R : w \in \Sigma^* \}$



- Automata "REMEMBERS" what word it has already read by putting it on STACK (one symbol at the time)
- Then it can be REMOVED from the stack and checked against the INPUT.

WORD ACCEPTED: END finite state + STACK EMPTY.

CF grammar G

$L = L(G) = \{ w w^R : w \in \Sigma^* \}$

$G = (V, \Sigma, R, S)$

$\Sigma = \{a, b\}$ $\quad V - \Sigma = S$

$R = \{ S \to aSa \mid bSb \mid e \}$

S

aSa

$abSba$

$abbSbba$

$abbaSabba$

$abbabSbabba$

$abbab \mid babba \in L(G)$

IDEA

Automata will read abbab putting babba on the stack (from DOWN-UP) STOP NONDETERMINISTICALLY and START to COMPARE the stack with the rest of the input.

IDEA

For REGULAR grammar we constructed the FA as follows:

$A \to wB$ then $(A, w, B) \in \Delta$

$A \to w$

In state A reads w moves to B

What about a rule

$A \to wBX$ $\quad w, x \in \Sigma^*$

We translate it to PUSH-DOWN AUTOM. as follows:

M in state A reads w,

PUT X ON THE STACK

goes to the state B

Later, the symbols on the stack can be removed and checked against the input.

THEOREM: The class of languages accepted by PDA is exactly the class of CF-languages

GOAL

DEFINITION (Push-down automaton)

$$M = (K, \Sigma, \Gamma, \Delta, s, F)$$

- K finite set (of STATES)
- Σ alphabet (of INPUT SYMBOLS)
- Γ alphabet (of STACK SYMBOLS)
- $s \in K$ (INITIAL state)
- $F \subseteq K$ (FINAL STATES)
- Δ – TRANSITION RELATION

$$\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$$

- Δ is FINITE

$((p, u, \beta), (q, \gamma)) \in \Delta$ means:

M in STATE p with β on the TOP of the STACK may read u from input tape, REPLACE β by γ on TOP OF the stack and q_0 to q.

$A \subseteq B = A \in B$

$$\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$$

state word stack state stack

$$((p, u, \beta), (q, \gamma)) \in \Delta$$

We use symbols β, γ for Γ^* STACK

Instruction:

- STATE p read u (from tape)
- GO TO STATE q AND
- REPLACE β by γ on THE TOP of the stack • go to q

PDA is NON-DETERMINISTIC

(Δ is a RELATION, not a function)

SOME CF languages can be accepted by the deterministic PDA (Δ is a FUNCTION) but DPDA $\neq$ PDA in general.

IDEA

- A PDA is deterministic (DPDA) if there is at most one transition applicable to each configuration. Δ is FUNCTION.
- A Language is a deterministic context-free if it is recognized by a DPDA automaton that also has the extra capability of sensing the END of the input string. DEF ≠ ♦CF

$$L = \{a^{m_1} b a^{m_2} b \dots a^{m_n} : n \geq 2,\ m_1 \dots m_n \geq 0 \text{ and } m_i \neq m_j \text{ for some } i, j\}$$

is a non-deterministic CF.

FOR MOST programming languages one can construct a DPDA that accepts ALL SYNTACTICALLY CORRECT PROGRAMS

Denise
- rejects programs with synt. errors
- provides mechanical syntactic analysis of programs; STAGE OF COMPILATION!

$M = (K, \Sigma, \Gamma, \Delta, s, F)$ PDA

$\Delta \in (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$

REPLACE β by γ on TOP OF the stack

$((p, u, \beta)(q, \gamma)) \in \Delta$ TRANSITION

$\beta \in \Gamma^*$

Special transitions

- PUSH a – add a to the top of the stack

$((p, u, e), (q, a))$ PUSHES a

- POP a – remove a from the top of the stack

$((p, u, a), (q, e))$ POPS a

a on the TOP of the stack

remove a from the TOP of the stack

- CONFIGURATION

$$(q, w, \gamma) \in K \times \Sigma^* \times \Gamma^*$$

q: state; w: input to be read; γ: content of the stack read top-down

(in FA: configuration $(p, w) \in K \times \Sigma^*$)

- TRANSITION RELATION

$$\vdash_M \subseteq (K \times \Sigma^* \times \Gamma^*)^2$$

Δ replaces β by γ on TOP of α

$$(p, ux, \beta\alpha) \vdash_M (q, x, \gamma\alpha)$$

iff

$$((p, u, \beta), (q, \gamma)) \in \Delta$$ (there is such elem in Δ)

$x \in \Sigma^*, \gamma, \alpha, \beta \in \Gamma^*$

$\vdash_M^*$ is reflexive, transitive closure of $\vdash_M$

$$L(M) = \{ w \in \Sigma^* : (S, w, e) \vdash_M^* (p, e, e) \text{ for certain } p \in F \}$$

$$L(M) = \{ w \in \Sigma^* : \exists_{p \in F} (S, w, e) \vdash_M^* (p, e, e) \}$$

$p \in F$

$(S, w, e) \xrightarrow[\text{apply rules}]{\vdash_M^*} (p, e, e)$

START with w and STOCK empty $\vdash_M^*$ FINISH in end state after reading w and EMPTYING ~~stuff~~ Stack.

M accepts $w \in \Sigma^*$ iff $w \in L(M)$

REMARK

Every FINITE AUTOMATON is a PUSH-DOWN Automaton (that never operates on its stack)

proof $M = (K, \Sigma, \Delta, s, F)$ a FA.

We construct

$$M' = (K, \Sigma, \Gamma, \Delta', s, F)$$

where $\Gamma = \phi$ $\quad e \in \Gamma^* = \phi^*$

$$\Delta' = \{((p, u, e), (q, e)) : (p, u, q) \in \Delta\}$$

$$L(M) = L(M')$$

FA $\subseteq$ PDA

EXAMPLES OF PDA.

Special (useful) transitions

- $((p, u, e), (q, a))$ PUSH a
- $((p, u, a), (q, e))$ POP a

In particular

$((p, a, a)(q, e))$ means

Read (a) when (a) is on the TOP of the stack and remove (a) from the top of the stack

COMPARE (a) on INPUT with the content of the stack.

Example 1

$G = (V, \Sigma, R, S)$

$V = \{S, a, b, c\}$, $\Sigma = \{a, b, c\}$

$R = \{ S \to aSa \mid bSb \mid c \}$

↓ c is a MARKER of the middle

$$L(G) = \{ \omega c \omega^R : \omega \in \{a, b\}^* \}$$

CONSTRUCT PD M, such that

$$L(M) = L(G)$$

$M = (K, \Sigma, \Gamma, \Delta, s, F)$

$K = \{s, f\}$ $\Sigma = \{a, b, c\}$

$\Gamma = \{a, b\}$ $F = \{f\}$

$\Delta = \{$

① $((s, a, e), (s, a))$ PUSH a (on the top of the stack)

② $((s, b, e), (s, b))$ PUSH b

③ $((s, c, e), (f, e))$ GO TO state $f \in F$ when see c

$\}$

④ $((f, a, a), (f, e))$ COMPARE a ON INPUT with a on stack and POP a

⑤ $((f, b, b), (f, e))\}$ COMPARE and POP b

Transitions (instructions) of

Δ :
- ① + ② Put what we are reading from INPUT on stack until we reach c (in inverse order!)
- ③ switches M, when it sees c to the state $f \in F$ (leaves stack untouched)
- ④ + ⑤ COMPARES input after c with what is on STACK (w) and removes when it MATCHES (all in state f)

$$L(M) = \{wcw^R : w \in \{a, b, \}^*\}$$

$abbcbba \in L(M)$

		stock		
s	abbcbba	e	①	Reads a + PUSH a
s	bbcbba	a	②	Reads b PUSH b
s	bcbba	ba	②	Reads b Push b
s	cbba	bba	③	Reads c switch to f
f	bba	bba	⑤	MATCHES b + removes
f	ba	ba	⑤	Matches b + removes
f	a	a	④	Matches and removes a
f	e	e		

$$(s, abbcbba, e) \xrightarrow[M]{*} (f, e, e)$$

Example 2

$G = (V, Z, R, S)$ $\Sigma = \{a, b\}$

$R = \{ S \to aSa \mid bSb \mid e \}$

$$L(G) = \{ ww^R : \; w \in \{a,b\}^* \}$$

CONSTRUCT PDA M such that

$$L(G) = L(M)$$

M construction goes as in the previous example for $L = \{wcw^R : w \in \Sigma^*\}$ but now the SWITCH to f is NON-DETERMINISTIC (we don't have the "middle" c) i.e

we replace $((s, c, e)(f, e)) \in \Delta_M$

by a NON-deterministic

$((s, e, e)(f, e))$: says: switch to f anytime

Many computations will not accept ww^R, but ONE would do!

best switch

$abaaba \in L(G) = L$

Find computation of PDA accepting this word.

NEW (replacement)

$((s, e, e), (f, e)) \in \Delta$

$[\ (s, c, e), (f, e)\ ) \in \Delta\ )$

	s	abaaba	e } NON accept
①	f	abaaba	e } end comp
		baaba	a end
	s	aaba	ba
	s	aaba	ba end
②	f		
	s	aba	aba
	f	aba	aba
	f	ba	ba) ok.
	f	a	a
	f	e	e

(c)

aaba babb = w

Show that $w \notin L(M)$

s aababaabb
① f aababaabb — e, e — (—)

s abababb — a
② f (a) bababb — a — (a)
f baba bb — e — stop (—)

s

You must cover all possible "searches"

and show that never set a path ending with (f, e, e)

$w \notin L(M)$

Example 3

Construct a PD A accepting

$$L = \{ w \in \{a,b\}^+ : \#a = \#b \}$$

IDEA:

M has a symbol (C) $\in \Gamma$ on the bottom of the stack as a MARKER

Either a's or b's are kept on the stack

A stack of a's indicate the EXCESS of a's over b's thus read, and vice-versa

b aabaabb $\in L$

$L_1 = \{a^n b^n : n \in \mathbb{N}\} \subset L$

$L = \{ w \in \{a,b\}^* : \#a = \#b \}$

$M = (K, \Sigma, \Gamma, \Delta, s, F)$

$K = \{s, q, f\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, c\}$,

$F = \{f\}$

$\Delta = \{$

① $((s, e, e), (q, c))$ 1. C on the bottom

② $((q, a, c), (q, ac))$ 2. Reads a and puts a on the bottom

③ $((q, a, a), (q, aa))$ 3. Reads a and pushes a on stack of a's

④ $((q, b, a), (q, e))$ 4. Pop a, when reads b

⑤ $((q, b, c), (q, bc))$ 5. puts b on bottom

⑥ $((q, b, b), (q, bb))$ 6. pushes b on stack of B's

⑦ $((q, a, b), (q, e))$ 7. Pops b, when reading a

⑧ $((q, e, c), (f, e))$ 8. All read, c only on the stack remove C.

$\}$

$e \in L(M)$ ① + ⑧

abbbaa bababa

(s)	abbbaa	(c)
q	bbbaa	(@c) c q #4
q	bbaa	c #5
q	baa	bc #6 (b c)
q	aa	bbc ((q, a, b)(q, e)) #7 ← Δ
q	a	bc
q	e	c

(q, e, c) → switch (f, e)

f e e

accepting computation.

We know

REG GRAMMAR ≡ FA

↓ proves regular language

↖ proves

Now we want to have similar connection between

CFGRAMMARS ≡ PDA

↓ means

(1) For each CF grammar G there is a PDA M such that L(G) = L(M)

(2) Each PDA recognizes a CF language

HARDER

Theorem 1

Each context-free language is accepted by some PDA automaton

Proof.

Let $G = (V, \Sigma, R, S)$ be a CFG.

we are going to define a PDA M such that $L(G) = L(M)$

M has two states p, q (start, final) and REMAINS permanently in state q (FINAL) after FIRST move.

M uses V as STACK alphabet (NONTERMINALS $(V-\Sigma)$)

$M = (K, \Sigma, \Gamma, \Delta, s, F)$

$K = \{p, q\}$, $\Sigma = \Sigma$, $\Gamma = V$

$s = p$, $F = \{q\}$

① M starts by PUSHING S on STACK and enters $q \in F$ ①

In each next step it either

- replaces the top symbol $A \in V$ on stack by x ② (for a rule $(A \to x) \in R$) $x \in (\Sigma \cup V)^*$ or
- POPS top symbol on stack that MATCHES the next input symbol ③

$$\Delta = \{ \text{①} ((p, e, e), (q, S)) \} \cup$$

Initial aut

start grammar

$$\text{②} \{ ((q, e, A), (q, x)) : \text{each } A \to x \in R \} \cup$$

$$\text{③} \{ ((q, a, a), (q, e)) : a \in \Sigma \}$$

Example

$G = (\{S, a, b, c\}, \{a, b\}, R, S)$

$R = \{S \to aSa \mid bSb \mid c\}$

$L(G) = \{\omega c \omega^R : \omega \in \Sigma^*\}$

PD AUTOMATA

$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$ — initial

$\Delta = \{$

① $((p, e, e), (q, S))$ — introd. S in gramm.

② GRAMMAR

$((q, e, S), (q, aSa))$ — $S \to aSa$

$((q, e, S), (q, bSb))$ — $S \to bSb$

$((q, e, S), (q, c))$ — $S \to c$

③ BUILDING Emptying the STOCK

$((q, a, a), (q, e))$

$((q, b, b), (q, e))$

$((q, c, c), (q, e))\}$

Show $abbcbba \in L(M)$

STATE	UN-read input	stack	transition
start p	abbcbba	e	1 pick $S \to aSa$
q	abbcbba	S	2 (non determ.)
q	abbcbba	aSa	3 Match a remove a
q	bbcbba	Sa	2 NON-DETERM. we pick $(q, \varepsilon, S)(q, bSb)$
q	bbcbba	bSba	3 Match/remove b
q	bcbba	Sba	2 NON-DET we pick
q	bcbba	bSbba	3 b
q	cbba	Sbba	2 $((q,\varepsilon,S),(q,c))$ $S \to c$
q	cbba	cbba	3
q	bba	bba	3 match + remove
q	ba	ba	3
q	a	a	3
q_a	e	e	END

abca $\in L(M)$ — REMARK

generate

ALL possibilities

and show that none

leads to (q, e) q - FINAL

(p, e)

final.

$S \to aSa$
$S \to bSb$
$S \to c$

p abca

2 abca

S

bSa → @

STOP

aSa

/

dbca

bca

aSa → 3 choices

Sa

show that we never get e e

Theorem 2

If a language is accepted by a PDA, then is CF.

PDA → GF grammars

p. 139-42

Theorem

The class of languages accepted by PDA is exactly the class of CF languages

(this is Theorem 1 + Theorem 2)

CFG ≡ PDA

Regular languages ≡ { DFA, NONDFA } FA, REG. GRAMMARS (special grams)

Context free languages ≡ { PDA, CF grammars }

DPDA ≠ CF

To prove that

L is a CFree we have devises

① Construct a CF grammar G

$L = L(G)$

or

② Construct PDA M

$L = L(M)$

We constr. PDA M for

$L = \{ w : \#a = \#b \}$

This proves

L is CFree ///

PROPERTIES OF CFL

(1) CLOSURE PROPERTIES

Theorem 1

The context-free languages are closed under $\cup$, $\circ$, * (union, concatenation, Kleene's star) (ONLY)

Proof (UNION)

$G_1 = (V_1, \Sigma_1, R_1, S_1)$, $G_2 = (V_2, \Sigma_2, R_2, S_2)$

assume $V_1 - \Sigma_1 \neq V_2 - \Sigma_2$

$G = G_1 \cup G_2 = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2,$

$R_1 \cup R_2 \cup \{S \to S_1 \mid S_2\}, S)$ ← new state

Take $w \in \Sigma_1 \cup \Sigma_2$

$S \overset{*}{\underset{G}{\Rightarrow}} w$ if $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ or $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$

$S \to S_1 \mid S_2$

{ Disjoint set of non terminals of G_1, G_2 — SD

$S_1 \overset{*}{\underset{G}{\Rightarrow}} w$ iff $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ and similar

$S_2 \overset{*}{\underset{G}{\Rightarrow}} w$ iff $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$, hence

$S \overset{*}{\underset{G}{\Rightarrow}} w$ iff $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} w$ OR $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} w$.

and $L(G) = L(G_1) \cup L(G_2)$

CONCATENATION $G = G_1 \circ G_2$

NEW START SYMBOL

$G = (V_1 \cup V_2 \cup \{S\},\ (\Sigma_1 \cup \Sigma_2),$

$R_1 \cup R_2 \cup \{ S \to S_1 S_2 \},\ S)$

KLEENE'S STAR $G = (V, \Sigma, R, S)$

$G^* = (V, \Sigma, R \cup \{S \to e, S \to SS\}, S)$

$e \in L^*$

Thm

The class of context-free languages is NOT CLOSED under intersection and complementation

Proof based on the fact (to be proved later) that

$L = \{a^n b^n c^n : n \geq 0\}$ is NOT CONTEXT-FREE

Need pumping lemma for CFL.

Take $L_1 = \{a^n b^n c^m : m, n \geq 0\}$ CF

$L_2 = \{a^m b^n c^n : n, m \geq 0\}$ CF

L_1, L_2 are CONTEXT-free

$L = L_1 \cap L_2$ is not context-free.

COMPLEMENTATION

L_1, L_2 CF Languages. Assume it is closed under complementation then

$$\underbrace{\Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))}_{CF} \text{ is a CFL.}$$

($\Sigma^* - L_1$: CF; $\Sigma^* - L_2$: CF)

but

$$\Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2)) = L_1 \cap L_2 \notin CFL$$

(de Morgan)

contradiction with the fact that CFL. are NOT closed under $\cap$.

Reg $\subseteq$ CF

THEOREM

IF L_1 is CF, L_2 is Regular

THEN $L_1 \cap L_2$ is CF

The intersection of a CF language with a regular is a CF language

Proof: CONSTRUCT a PD automata

$CF \cap REG = CF$

Proof (Intersection of CF and R is CF)

$L_1 = L(M_1)$ $M_1 = (K_1, \Sigma_1, \Gamma_1, \Delta_1, s_1, F_1)$ CF

push down automata

LET $L_2 = L(M_2)$ $M_2 = (K_2, \Sigma_2, \delta_2, s_2, F_2)$ DFA

IDEA Combine M_1, M_2 into a PDA M that carries computations of $M_1 M_2$ IN PARALLEL and accepts w only when $M_1 M_2$ does.

$M = (K, \Sigma, \Gamma, \Delta, s, F)$ $M = M_1 \times M_2$

$K = K_1 \times K_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$

$\Gamma = \Gamma_1$, $s = (s_1, s_2)$, $F = F_1 \times F_2$

Δ is defined as follows:

$(((q_1, q_2), u, \beta), ((p_1 p_2), \gamma)) \in \Delta$

iff

$((q_1, u, \beta)(p_1, \gamma)) \in \Delta_1$ and

$(q_2, u) \overset{*}{\vdash}_{M_2} (p_2, e)$ $u \in L(M_2)$ when

PROOF

$L(M) = L(M_1) \cap L(M_2)$ C.F.

M_2 READS u.

$$L = \{ wcw : w \in \Sigma^* \}$$

[?]